

Spotlight the Negatives: A Generalized Discriminative Latent Model

Hossein Azizpour¹
azizpour@kth.se

Mostafa Arefiyan²
mostafa@brown.edu

Sobhan Naderi Parizi²
sobhan@brown.edu

Stefan Carlsson¹
stefanc@csc.kth.se

¹ CVAP
Royal Institute of Technology (KTH)
Stockholm, Sweden

² LEMS
Brown University
Providence, USA

Abstract

Discriminative latent variable models (LVM) are frequently applied to various visual recognition tasks. In these systems the latent (hidden) variables provide a formalism for modeling structured variation of visual features. Conventionally, latent variables are defined on the variation of the foreground (positive) class. In this work we augment LVMs to include *negative* latent variables corresponding to the background class. We formalize the scoring function of such a generalized LVM (GLVM). Then we discuss a framework for learning a model based on the GLVM scoring function. We theoretically showcase how some of the current visual recognition methods can benefit from this generalization. Finally, we experiment on a generalized form of Deformable Part Models with negative latent variables and show significant improvements on two different detection tasks.

Introduction

Discriminative latent variable models (LVM) are frequently and successfully applied to various visual recognition tasks [6, 7, 10, 13, 12, 15, 16]. In these systems the latent (hidden) variables provide a formalism for modeling structured variation of visual features. These variations can be the result of different possible object locations, deformations, viewpoint, subcategories, etc. Conventionally, these have all been defined only based on the foreground (positive) class. We call such latent variables “positive”. In this work we introduce generalized discriminative LVM (GLVM) which use both “positive” and “negative” latent variables. Negative latent variables are defined on the background (negative) class and provide *counter evidence* for presence of the foreground class. They can, for instance, learn mutual exclusion constraints, model scene subcategories where the positive object class is unlikely to be found, or capture specific parts which potentially indicate the presence of an object of a similar but not the same class.

The objective of the proposed framework is to learn a model which maximizes the evidence (characterized by positive latent variables) for the foreground class and at the same time minimizes the counter evidence (characterized by negative latent variables) lying in background class. Thereby GLVM empowers the latent variable modeling by highlighting the role of

negative data in its own right. An interesting analogy is with game theory; when playing a game a simple strategy is to maximize your score. This resembles LVM's objective which is to maximize evidence for the presence of the foreground class. A better strategy, however, is to maximize your scores at the same time as minimizing the opponents' scores. This is analogous to GLVM's objective which takes into account counter evidence emerging from negative latent variables while looking for evidence from the foreground class. The score of the opponents is counter evidence in our hypothetical example.

The concept of *negative parts* was noted in [10]. However, [10] focuses on automatic discovery and optimization of a part based model with negative parts. In this paper we extend the notion of negative parts to negative latent variables and propose a framework for defining models that use both positive and negative latent variables.

Many different modeling techniques benefit from latent variables. Felzenszwalb *et al.* [6] introduced latent SVM to train deformable part models, a state of the art object detector [7]. Yu and Joachims [16] transferred the idea to structural SVMs. Yang *et al.* [15] proposed a kernelized version of latent SVM and latent structural SVM and applied it to various visual recognition tasks. Razavi *et al.* [13] introduced latent variables to the Hough transform and achieved significant improvements over a Hough transform object detection baseline. And-Or trees use latent variable modeling for object recognition and occlusion reasoning [14].

In this paper we introduce a novel family of models which generalizes discriminative latent variable models. We review LVMs in Section 2, formulate our generalization of LVMs in Section 3. Then, we discuss training of GLVMs and propose an algorithm for learning the parameters of a GLVM in Section 4. In Section 6 we discuss how various computer vision models can benefit from GLVM. Finally, in Section 7 we experiment on generalized DPMs.

2 Discriminative Latent Variable Models (LVM)

Consider a supervised classification problem provided with a set of n training examples $X \in \mathcal{R}^{d \times n}$ and their corresponding labels $Y \in \mathcal{Y}^n$. The goal of discriminative learning is then to learn a scoring function $S_w(x, y)$, parametrized by w , that ranks a data point x_i and its true label y_i higher than the rest of labels. For a linear scoring function this is written as follows:

$$S_w(x_i, y_i) > S_w(x_i, y) \quad \forall y \neq y_i \quad (1)$$

$$S_w(x_i, y) = w^T \phi(x_i, y) \quad (2)$$

Here $\phi(x, y)$ is the representation function (e.g. Histogram of Oriented Gradients, ConvNet representation, etc.). Since the scoring function of (2) is linear in w any invariance to non-linear object class deformation should be reflected in the representation function ϕ . This requires a very rich feature encoding. LVMs model known structured variations of the object class by using various latent variables a corresponding to different sources of variation. Each latent variable models a source of variation and accordingly alters the representation ϕ so that it best matches the model parameters w . Thus, all latent variables in an LVM are maximized over. For instance, in weakly supervised object detection location of the object in an image is modeled by a latent variable. The scoring function of an LVM takes this form:

$$S_w(x_i, y) = \max_a w^T \phi(x_i, y, a) \quad (3)$$

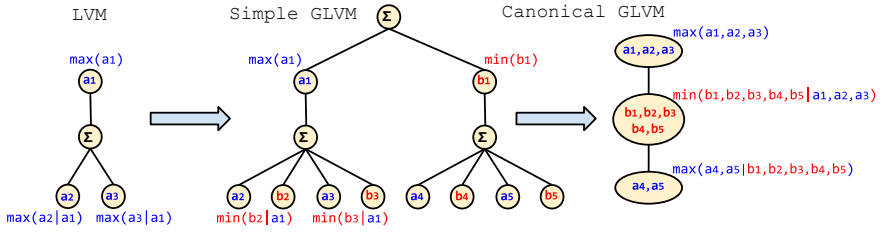


Figure 1: Example of an LVM scoring function on three latent variables (left), a Simple GLVM scoring function (middle), and a GLVM in its canonical form. The models become more and more complex from left to right.

In some cases the set of labels \mathcal{Y} comprises of only known classes. In open-set classification, however, one of the possible labels in \mathcal{Y} refers to an *unknown class*. The simplest form of open-set classification is binary classification where samples are classified into the class of interest where $y = +1$ (e.g. cat) or anything else where $y = -1$ (e.g. non-cat) which are so called *foreground* and *background* respectively. In this work we are interested in open-set classification problems which suit many real-world applications better. In the next section we show how the latent variables in the scoring function of (3) can be altered to “model” the negative data in its own right and thereby enrich the expressive power of discriminative latent variable models. In the rest of the paper we focus on the binary classification problem.

3 Generalized LVM with Negative Modeling (GLVM)

The scoring function in discriminative LVM is usually constructed in such a way that it looks for “evidence” for the foreground class, and the weakness (or lack) of such evidence indicates that the input is from the background class. For instance, in detecting *office* scenes, the location of a work desk (evidence for office) can be modeled by a latent variable.

We generalize LVMs by equipping them with latent variables that look for “counter evidence” for the foreground class. For example, detecting a stove (evidence for kitchen) or sofa (evidence for living room) provides counter evidence for office. Such latent variables can also be contextual, for example for a *cow* detector, presence of saddle/rider/aeroplane counts as counter evidence whereas presence of meadow/stable/milking parlor counts as evidence. For visual examples refer to Section 1 in supplementary material.

We call the latent variables that collect evidence for the foreground class *positive latent variables* and the latent variables that collect counter evidence for the foreground class *negative latent variables* and denote them by a and b respectively throughout the paper.

Representation function. Before we go further, let’s explain how our representation function ϕ is constructed. We assume the model parameters (associated with *all* latent variables) are concatenated into a single vector w . Moreover, w and its corresponding feature vector $\phi(x_i, a_1, b_1, a_2, b_2, \dots)$ have the same dimensionality. Every latent variable a_k or b_k encodes its own place in $\phi()$. The places for different latent variables do not overlap. When some latent variables are omitted from the inputs of $\phi()$, their place is filled with zero vectors. These assumption are made for the ease of notation in the upcoming formulations. For clarity we take three steps towards defining the generic form of GLVM scoring function.

Simple Form. The simplest form of a GLVM scoring function involving two independent sets of positive and negative latent variables is (\mathcal{Z} is the set of possible latent assignments):

$$S_w(x_i, y) = \max_{a \in \mathcal{Z}^+} w^T \phi(x_i, y, a) - \max_{b \in \mathcal{Z}^-} w^T \phi(x_i, y, b) \quad (4)$$

Note that the second max cannot be absorbed in the first max. Thus, the GLVM scoring function is more general than that of LVM. In particular, (4) reduces to (3) when $|\mathcal{Z}^-| \leq 1$. One can characterize positive and negative latent variables according to the way they impact the value of the scoring function. Each assignment of a latent variable has a corresponding value associated to it. For instance, in the deformable part models (DPMs) latent variables indicate the location of the object parts [6]. Each assignment of a latent variable in a DPM corresponds to a placement of a part which in turn is associated with a value in the score map of the part. A positive (negative) latent variable is one whose associated value is positively (negatively) correlated with the final value of the scoring function. This should not be confused with negative entries in the model vector w . In a linear model with the score function $S(x) = w^T \phi(x)$ the value of the d -th dimension of the feature vector (i.e. $\phi(x)_d$) is negatively correlated with $S(x)$ when $w_d < 0$. Yet, negative latent variables are fundamentally different from this. In particular, a negative latent variable cannot be obtained by flipping the sign in some dimensions of the parameter vector and using a positive latent variable instead. The value associated with a negative latent variable (i.e. $\max_b w^T \phi(x, b)$) is a convex function of w whereas the value associated to negative entries in w (i.e. $w_d \phi(x)_d$) is linear in w .

Dependent Form. The value of the GLVM score can be thought of as the value of the best strategy in a 2-player game. To show this we need to modify the notation as follows:

$$\begin{aligned} S_w(x_i) &= \max_{a \in \mathcal{Z}^+} w^T \phi(x_i, a) + \min_{b \in \mathcal{Z}^-} -w^T \phi(x_i, b) \\ &= \max_{a \in \mathcal{Z}^+} \min_{b \in \mathcal{Z}^-} w^T \phi(x_i, a) - w^T \phi(x_i, b) \\ &= \max_{a \in \mathcal{Z}^+} \min_{b \in \mathcal{Z}^-} w^T \phi(x_i, a, b) \end{aligned} \quad (5)$$

In the first step we dropped y from the notation since we will be focusing on binary classification. In the second step min is pulled back. For the last step we let $\phi(x_i, a, b) = \phi(x_i, a) - \phi(x_i, b)$ although any feature function that *depends on both* b and a would do. So, (5) is more general than (4). Note that the game theory analogy of minimax is now more apparent in the new formulation of (5).

Canonical Form. The scoring function in (5) is linear when the latent variables a and b are fixed. One can imagine augmenting this linear function with more positive and negative latent variables; the same way that we extended (2) to get to (5). This creates multiple *interleaved* negative and positive latent variables. This way, we can make a hierarchy of alternating latent variables recursively. This general formulation has interesting ties to compositional models, like object grammar [6], and high-level scene understanding. For example, we can model scenes according to the presence of some objects (positive latent variable) and the absence of some other objects (negative latent variable). Each of those objects themselves can be modeled according to presence of some parts (positive) and absence of others (negative). We can continue the recursion even further. The GLVM scoring function, after being expanded recursively, can be written in the following general form:

$$S_w(x_i) = \max_{a_1} \min_{b_1} \max_{a_2} \min_{b_2} \dots \max_{a_K} \min_{b_K} w^T \phi(x_i, (a_k, b_k)_{k=1}^K) \quad (6)$$

We call this the *canonical form* of a GLVM scoring function. Note that any other scoring function that is a linear combination of max's and min's of linear functions can be converted to this canonical form by simple algebraic operations. Moreover, (6) with $K > 1$ and non empty a_k and b_k cannot be simplified into (5), in general. Its proof follows the *max-min inequality* [11]. See Figure 1 for an example.

4 Training GLVMs for Classification

In this section we propose an algorithm for training GLVMs to do binary classification. We focus on SVM training objective and use hinge loss. That being said, the same idea can be used to optimize any training objective that involves a linear combination of convex and/or concave functions of GLVM scores; *e.g.* SVM with squared hinge loss. For the ease of notation we drop the subscript w in S_w in the rest of this section.

Let $D = \{(x_i, y_i)\}_{i=1}^n$ denote a set of n labeled training examples. Each training example is a pair (x_i, y_i) where x_i is the input data and $y_i \in \{+1, -1\}$ is the ground-truth label associated with it. SVM training objective can be written as follows:

$$O(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i S(x_i)) \quad (7)$$

where $S(x_i)$ is a scoring function. Training a classifier requires finding the model w^* that minimizes the function *i.e.* $w^* = \arg \min_w O(w)$. In most practical applications, however, the objective function O is complex (in particular, non-convex). This makes it hard to find the global minimizer. Thus, we usually have to resort to local minima of the objective function.

The concave-convex procedure (CCCP) [14] is a general framework for optimizing non-convex functions. For minimization, in each iteration the concave part is replaced by a hyperplane tangent to it at the current solution. The result is a convex function. Local minima of a convex function are global minima. Thus, one can use simple methods such as gradient descent to minimize a convex function.

We train GLVMs using an iterative algorithm that is very similar to CCCP [14]. But, unlike CCCP, we do not linearize the concave part of the objective function. Our training algorithm alternates between two steps: 1) constructing a convex upper bound to the training objective O and 2) minimizing the bound. Let $B_t(w)$ denote the bound constructed at iteration t . Also let $w_t = \arg \min_w B_t(w)$ be the minimizer of the bound. All bounds must touch the objective function at the solution of the previous bound; that is $B_t(w_{t-1}) = O(w_{t-1}), \forall t$. This process is guaranteed to converge to a local minimum or a saddle point of the training objective O .

To construct bounds B_t , as we shall see next, we only need to construct a concave lower bound $\hat{S}(x_i)$ and a convex upper bound $\check{S}(x_i)$ to the scoring function of (6). We define B_t as follows:

$$B_t(w) = \frac{1}{2} \|w\|^2 + C \sum_{\substack{i=1 \\ y_i=-1}}^n \max(0, 1 + \check{S}_t(x_i)) + C \sum_{\substack{i=1 \\ y_i=+1}}^n \max(0, 1 - \hat{S}_t(x_i)) \quad (8)$$

In the next section we will explain how $\check{S}(x_i)$ and $\hat{S}(x_i)$ are constructed for GLVMs. Algorithm 1 in the supplementary material summarizes all steps in our training procedure.

Convex Upper Bound and Concave Lower Bound. Here we explain how we obtain $\check{S}_t(x_i)$ and $\hat{S}_t(x_i)$ when GLVM scoring function (6) is used. We fix all the negative latent variables in $S(x_i)$ to get $\check{S}_t(x_i)$ and fix all the positive latent variables to get $\hat{S}_t(x_i)$.

Let w_t denote the model at iteration t and let $a = (a_1, \dots, a_K)$ be an arbitrary assignment of the positive latent variables. Given a , we denote the fixed values for the negative latent

variables on image x_i by $b^{(i)}(a) = (b_1^{(i)}, \dots, b_K^{(i)})$ and define it as in (9). Similarly, we define $a^{(i)}(b) = (a_1^{(i)}, \dots, a_K^{(i)})$ as in (10).

$$b^{(i)}(a) = \arg \min_{b_1, \dots, b_K} w_t^T \phi \left(x_i, (a_k, b_k)_{k=1}^K \right) \quad (9)$$

$$a^{(i)}(b) = \arg \max_{a_1, \dots, a_K} w_t^T \phi \left(x_i, (a_k, b_k)_{k=1}^K \right) \quad (10)$$

Note that (9) uses a min and (10) uses a max. Also, note that in both equations the fixed assignments are computed according to w_t . Finally, we define $\check{S}_t(x_i)$ and $\hat{S}_t(x_i)$ as follows:

$$\check{S}_t(x_i) = \max_{a_1} \max_{a_2} \dots \max_{a_K} w_t^T \phi \left(x_i, \left(a_k, b_k^{(i)}(a) \right)_{k=1}^K \right) \quad (11)$$

$$\hat{S}_t(x_i) = \min_{b_1} \min_{b_2} \dots \min_{b_K} w_t^T \phi \left(x_i, \left(a_k^{(i)}(b), b_k \right)_{k=1}^K \right) \quad (12)$$

(11) is convex because it is a max of linear functions and it is an upper bound to $S(x)$ because the min functions are replaced with a fixed value. Similarly for (12).

For discussions regarding the dependency structure of the latent variables and how it affects the scoring and inference functions refer to Section 2 in the supplementary material.

5 Connection to existing Models

LSSVM: An interesting connection is to latent structural SVMs (LSSVM) [16] where the scoring function is given by,

$$S(x_i, y_i) = \max_h w^T \phi(x_i, y_i, h) \quad (13)$$

In this form, the formulation of scoring function does not make use of negative latent variables as defined in GLVM. In LSSVM objective function the individual loss induced by a sample (x_i, y_i) is calculated as below,

$$L(x_i, y_i) = \max_{(y, h)} [w^T \phi(x_i, y, h) + \Delta(y_i, y, h)] - \max_h w^T \phi(x_i, y_i, h) \quad (14)$$

Note that the representation function takes as input both class y and latent variable h . Now, assume a binary classification ($y \in \{-1, 1\}$) framework with a symmetric loss function Δ independent of h . Then, the scoring and loss function can be reformulated as,

$$S'(x_i) = \max_h w^T \phi(x_i, +1, h) - \max_h w^T \phi(x_i, -1, h) \quad (15)$$

$$L(x_i, y_i) = \max(0, \Delta(+1, -1) - y_i S'(x_i)) \quad (16)$$

Note that (15) is similar to the shallow structure of GLVMs in (4). This shows how negative latent variables can potentially emerge from LSSVMs.

Mid-level features based recognition: Many recent scene image classification techniques pre-train multiple mid level features (parts) [8, 9, 9] to represent an image. They construct the representation of an image by concatenating the maximum response of each part in the image and then train a linear classifier on top of those representations. Naderi *et al.* [10] have shown that training a classifier on those representations effectively corresponds to the definition of *negative parts*. In these works, the score of a part at an image is *maximized* over

different locations and thus its position and scale are, in essence, treated as a latent variables. In that respect, a negative weight in the linear classifier for such a latent variable can be translated to a *negative latent variable* (i.e. its maximum score contributes negatively to the score of an image). The training procedure for the scene classification models of [9, 9, 9, 10] is similar to the shallow version of the GLVM framework proposed in this paper.

ConvNet: [9] argues that the kernels in Convolutional Networks (ConvNet) can be interpreted as parts in part based models and shows that DPM is effectively a ConvNet. However, since ConvNets have the extra ability of learning weights (potentially negative) over these multiple kernels it is modeling counter evidences in a similar form as GLVM.

6 Showcases

To elicit the implications of the proposed generalization of LVMs, in this section, we review a few different state of the art computer vision models and demonstrate how they can benefit from the proposed framework. We consider adding negative latent variables to deformable part models, And-Or trees, and latent Hough transform and discuss the implications of it.

Show Case 1: Deformable Part Models. Deformable Part Model (DPM) [6] is an object detector that models an object as a collection of deformable parts. DPM uses multiple mixture components to model viewpoint variations of the object. The location of the parts (denoted by l_j) and the choice of the mixture component (indexed by c) are unknown and thus treated as latent variables. The scoring function of a DPM with n parts can be written as follows:

$$S(x) = \max_{c=1}^k \sum_{j=1}^n \max_{l_j} w_{c,j}^T \phi(x, l_j, c) \quad (17)$$

We extend DPMs to include negative parts. We refer to this as generalized DPM (GDPM). For instance, when detecting cow, positive parts in DPM include head, back, legs of a cow. In GDPM one can imagine several candidates for negative parts. For example, since horses appear frequently in the high scoring false positives of cow detectors, saddle or horse-head may be good negative parts for a cow detector. We define GDPM scoring function as:

$$S(x) = \max_{c=1}^k \left[\sum_{j^+=1}^n \max_{l_{j^+}} w_{c,j^+}^T \phi(x, l_{j^+}, c) - \sum_{j^-=1}^m \max_{l_{j^-}} w_{c,j^-}^T \phi(x, l_{j^-}, c) \right] \quad (18)$$

Following the recursion of GLVM, the outer max can also be cloned with a negative counterpart. In that respect, the negative *components* would look, for instance, for different viewpoints of a horse with its own positive and negative parts. We implemented GDPM and evaluated it on two different datasets. We discuss the experimental results in Section 7.

Show Case 2: And-Or trees. And-Or trees [12] are hierarchical models that have been applied to various visual recognition tasks. And-Or trees are represented by a tree-structured graph. Edges in the graph represent dependencies and nodes represent three different operations. *Terminal* nodes (T_i) are directly applied to the input image. A typical example of a terminal node is a template applied at a specific location of the image. *Or* nodes (O_i) take the maximum value among their children. An *Or* node can model, for example, the placement of terminal nodes or the choice of mixture components. *And* nodes (A_i) aggregate information

by summing over their children. The scoring functions of these nodes are defined as follows:

$$S(T_i|I; w_i) = w_i^T I(T_i), \quad S(A_i|C) = \sum_{c \in C(A_i)} S(c), \quad S(O_i|C) = \max_{c \in C(O_i)} S(c) \quad (19)$$

C encodes the hierarchy of the nodes and $C(N)$ denotes children of node N . In order to compute the score of an input image the scores are propagated from the terminal nodes to the root of hierarchy using dynamic programming. Or nodes find the most plausible interpretation, And nodes aggregate information, and terminal nodes collect positive evidence for the foreground class. Generalizing an And-Or tree with negative latent variables introduces a new type of node which we call a *Nor* node (N_i). Nor nodes collect counter evidence for the class of interest. We define the scoring function of a Nor node as follows:

$$S(N_i|C) = \min_{c \in C(N_i)} S(c) \quad (20)$$

Show Case 3: Latent Hough Transform. Hough transform (HT) has been used as a non-parametric voting scheme to localize objects. In Implicit Shape Models [10] the score of a hypothesis h (e.g. object location and scale) in an image x is calculated by aggregating its compliance to each *positive* training images. Let D_p denote the set of positive training images. Also let $S(x, h|x_i)$ denote the vote of a positive training image $x_i \in D_p$ in favor of hypothesis h . The scoring function of HT is defined as follows:

$$S_{HT}(x, h) = \sum_{i \in D_p} S(x, h|x_i) \quad (21)$$

One of the main issues with HT is the aggregation of votes from inconsistent images. For instance, training images of side-view cars contribute to the voting in a test image of a frontal-view car. Razavi *et al.* [13] alleviate this issue by introducing latent Hough transform (LHT). LHT groups consistent images (e.g. exhibiting same viewpoints) together using a latent variable $z \in Z$. Let $\phi(x, h)$ denote the vector of votes for hypothesis h obtained from all positive training images *i.e.* $\phi(x, h)_i = S(x, h|x_i)$. LHT assigns weight $w_{z,i}$ to the positive training image $x_i \in D_p$ indicating its relevance to the selected mixture component z . Scoring function of an LHT parameterized by w is defined as follows:

$$S_{LHT}(x, h) = \max_{z \in Z} w_z^T \phi(x, h) \quad (22)$$

In order to reformulate the problem as a GLVM we need to introduce *negative votes* [14] cast by negative training images in the scoring function of HT.

$$S_{HT}(x, h) = \sum_{i \in D_p} S(h|x_i) - \sum_{i \in D_n} S(h|x_i) \quad (23)$$

Then the scoring function of a GLVM generalization of LHT is given by:

$$S_{LNHT}(x, h) = \max_{z^+} w_{z^+}^T \phi(x, h) - \max_{z^-} w_{z^-}^T \phi(x, h) \quad (24)$$

It should be emphasized that negative latent variables might be crucial when using negative votes in HT due the high multimodality of negative data distribution. That is negative images are probably more susceptible to the aggregation of inconsistent votes. This might explain the reason why negative voting is not popular for HT based object recognition. Note that the GLVM version of LHT can make use of the negative data in a more effective way.

	bird	cat	cow	dog	horse	sheep
DPM₈	10.9	16.6	21.9	12.5	56.0	18.0
GDPM₇¹	10.9	17.0	22.0	12.5	57.2	19.2
GDPM₆²	9.5	18.6	23.2	12.2	56.0	19.8
GDPM₅³	10.5	13.9	21.7	11.9	54.8	19.1

(a) AP for animal classes of VOC 2007, comparing DPM with GDPM with different number of parts. Sub (super) indices indicate the number of positive (negative) parts.

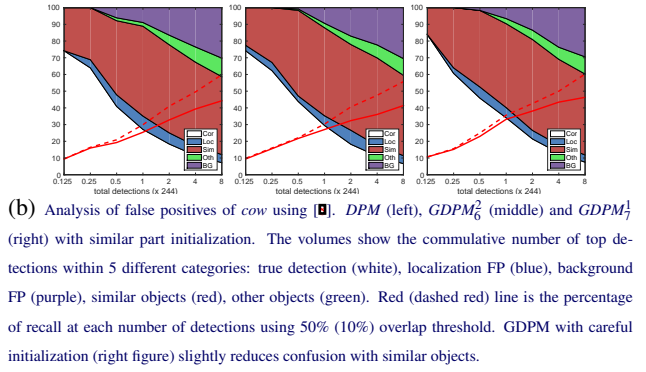



Figure 2: PASCAL VOC 2007

7 Experiments

To demonstrate the efficacy of a GLVM we evaluate it on deformable part models (DPM) by augmenting it with negative parts (GDPM). As discussed in Section 4 adding negative parts changes the optimization framework of DPMs. This includes many noteworthy details about inference, relabeling, data mining, features cache, etc. For details of these modifications refer to Section 4 in supplementary material. We conducted our experiments of GDPM on two different datasets. Similar to [1] HOG is used as feature descriptor but the observed trends should be independent of this choice. The results are measured using Average Precision of the PR-curve. We use the PASCAL VOC criterion of 50% intersection over union for *recall*.

PASCAL VOC 2007 Animals. We first evaluate GDPM on animal subset of VOC 2007. The training set of VOC 2007 contains about 4500 negative images and a few hundreds (~ 400) of positive instances per animal class. The same goes for the test set. We use the same hyper-parameters as used for original DPM [1], namely 6 components (3 + 3 mirror components) and 8 parts per component. Our results slightly differ from those reported in [1] due to absence of post-processing (bounding box prediction and context re-scoring) and slight differences in implementation. Figure 2a summarizes the results for substituting some positive parts in DPM with negative parts. 4 out of the 6 tested classes benefit from replacing positive parts with negative parts. Depending on the class the optimal number of negative parts is different. For PR curves refer to Section 6 in supplementary material.

Initialization. Similar to DPM [1], we initialized the location and appearance of positive (negative) parts by finding the sub-patches of the root filter which contained highest positive (negative) weights. However, we observed that negative parts in GDPM are quite sensitive to initialization. This is due to the non-convexity of the optimization. Furthermore, while the positive label corresponds to a single visual class the negative label encompasses more classes and thus exhibits a multi-modal distribution. But, we are interested in *discrimination* of the positives, thus we only need to model the boundary cases of the negative distribution. Studying these boundary cases for DPM model of *cow* revealed that most of negative samples come from "sheep" and "horse". Initializing GDPM negative parts of *cow* with a part from similar objects increased the performance from **21.9** to **26.7**.



	Abyssinian	Bengal	Birman	Bombay	British Shorthair	Egyptian Mau	Maine Coon	Persian	Ragdoll	Russian Blue	Siamese	Sphynx
DPM_4	21.3	12.8	34.5	23.3	32.2	15.8	21.6	28.0	19.4	24.0	29.4	22.7
DPM_6	22.2	12.8	31.4	21.5	31.3	16.5	26.0	29.0	20.6	25.0	30.9	22.0
$GDPM_2^2$	24.3	11.9	38.4	23.9	31.0	19.7	27.7	29.7	24.5	29.9	35.9	27.4
$GDPM_4^4$	25.5	13.7	34.0	23.0	33.5	20.9	24.5	30.0	21.9	25.3	30.6	23.2

Table 1: Cat Head Detection: Results comparing DPM and GDPM with different number of positive and negative parts. Sub (super) indices show the number of positive (negative) parts. GDPM consistently outperforms DPM variants. The average cat head images are taken from [13].

Negative Parts. Although, adding negative parts helps in general, it seems that, unless the parts are carefully initialized, the discovered negative parts with default initialization are not visually meaningful. We analyzed the false positives (FP) of DPM and GDPM detectors similar to [8] and observed that some of the FP reduction in GDPM is from non similar object classes or background. However, when we initialized the negative parts using other classes, the ratio of FP due to similar objects decreased more significantly (See Figure 2b).

Cat Head. The second task is detection of heads for 12 breeds of cats. The cat head images are taken from Oxford Pets dataset [14]. For each cat breed the distractor set contains full images (including background clutter) of the other 11 breeds and 25 breeds of dogs. Each cat breed has around 65 positive images and 3500 negative images for training. Test set has the same number of images. This task is particularly hard due to the low number of positive training images and high level of similarity between cats heads which is sometimes hard for humans to distinguish. For sample images refer to Section 5 in supplementary material. Table 1 reports the results by comparing DPM and GDPM using various number of positive/negative parts. The aligned average images in [15] is used as header in Table 1. Since the number of positive images in the training set is low, we used one component per detector. It can be seen that GDPM consistently outperform the DPM using different number of parts. For a few classes (e.g. Bengal) DPM with 4 positive parts outperform GDPM with 2 positive and 2 negative parts, but as we increase the number of parts to 6, replacing 2 positive parts with 2 negative parts proves beneficial. We observed that adding more parts (more than 6) degrades the results due to over-fitting to the small number of positive images. Negative parts seems to be more robust for the classes where going from total of 4 parts to 6 parts show signs of over-fitting (e.g. Birman, Sphynx).

8 Conclusion

In this work we proposed a new framework for discriminative latent variable models by explicitly modeling *counter evidences* associated with negative latent variables along with the positive evidences and by that emphasized the role of negative data. We further described a general learning framework and showcased how common computer vision latent models can benefit from the new perspective. Finally, we experimented on two tasks of object detection revealing the benefits of negative latent variables. We further observed that expressive power of GLVM crucially depends on a careful initialization.

We think the proposed GLVM framework yields to fruitful discussions and works regarding various latent variable models.

Acknowledgement. We’d like to thank Pedro Felzenszwalb for the fruitful discussions.

References

- [1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.
- [2] Christopher M. Brown. Inherent bias and noise in the hough transform. *PAMI*, 5(5):493–505, 1983.
- [3] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, pages 494–502, 2013.
- [4] Ian Endres, Kevin J. Shih, Johnston Jiaa, and Derek Hoiem. Learning collections of part models for object recognition. In *CVPR*, pages 939–946, 2013.
- [5] Pedro Felzenszwalb and David McAllester. Object detection grammars. In *Technical Report*, 2010.
- [6] Pedro F. Felzenszwalb, David A. McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.
- [7] Ross B. Girshick, Forrest N. Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. In *CVPR*, 2015.
- [8] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *ECCV*, pages 340–353, 2012.
- [9] Mayank Juneja, Andrea Vedaldi, C. V. Jawahar, and Andrew Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, pages 923–930, 2013.
- [10] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77(1-3):259–289, 2008.
- [11] Sobhan Naderi Parizi, Andrea Vedaldi, Andrew Zisserman, and Pedro F. Felzenszwalb. Automatic discovery and optimization of parts for image classification. In *ICLR*, 2015.
- [12] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [13] Nima Razavi, Juergen Gall, Pushmeet Kohli, and Luc J. Van Gool. Latent hough transform for object detection. In *ECCV*, pages 312–325, 2012.
- [14] Xi Song, Tianfu Wu, Yunde Jia, and Song-Chun Zhu. Discriminatively trained and-or tree models for object detection. In *CVPR*, pages 3278–3285, 2013.
- [15] Weilong Yang, Yang Wang, Arash Vahdat, and Greg Mori. Kernel latent SVM for visual recognition. In *NIPS*, pages 818–826, 2012.
- [16] Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *ICML*, pages 1169–1176, 2009.
- [17] Alan Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4): 915–936, 2003.
- [18] Jun-Yan Zhu, Yong Jae Lee, and Alexei A Efros. Averageexplorer: Interactive exploration and alignment of visual data collections. *SIGGRAPH*, 33(4), 2014.

Supplementary Material

1 Intuitive example of LVM and GLVM

Figure 1 gives intuitive examples of positive and negative latent variables.

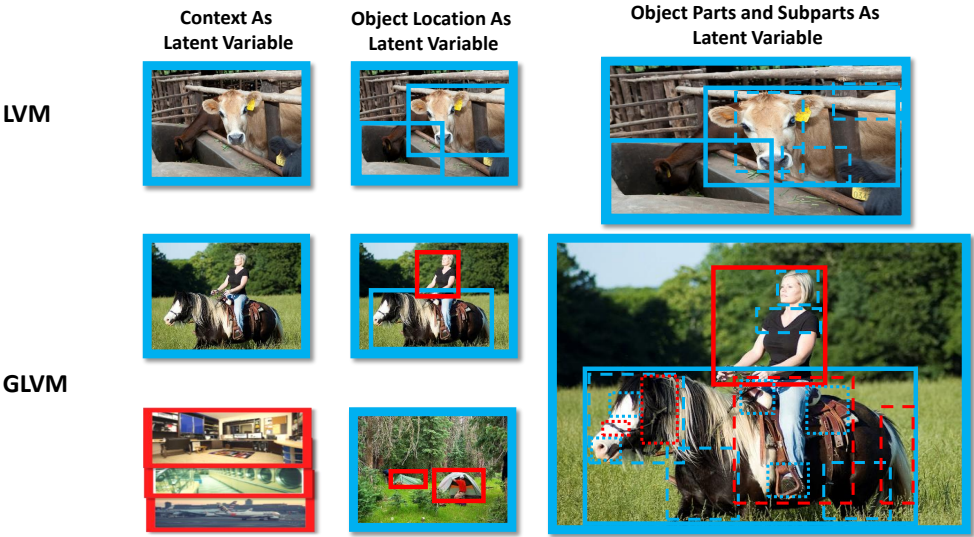


Figure 1: **Intuitive example of LVM and GLVM.** Assume the task of *cow* image classification. This figure visualizes examples of possible latent variables for LVM and GLVM models. A latent variable can be assigned at image level such as the background scene and then gradually go down a hierarchy of objects, parts and subparts. Positive latent variables (modeling evidences for cow) of an LVM may correspond to different scenes where cow is likely to be seen *e.g.* stable, meadow (blue boxes in the first column). Another positive latent variable can be assigned to the location of the cow in the image (second column, first row). In addition, location of different parts of cow *e.g.* head, back, legs can be modeled as other latent variables dependent to the latent variable of cow location (top right). All these examples of latent variables in LVMs look for evidences for the existence of cow. In GLVM we encourage the modeling of counter evidences through negative latent variables. For example at high level a negative latent variable may correspond to scenes where cow is unlikely to be found *e.g.* office, laundromat, or airport (bottom left). Meadow is an evidence for cow, but cows are less likely to be seen in a meadow beside camping tents (a dependent negative latent variable to positive latent variable meadow). Horses are visually similar to cow but people usually do not ride cows thus a human on top of cow can be negative evidence (a negative latent variable dependent to the positive latent variable of cow location). Furthermore, human, a negative latent variable for cow, can have its own positive latent variable as parts *e.g.* face and shoulders (bottom right).

2 Dependency Structure

Complexity of evaluating GLVM scoring function is governed by the dependency structure of the underlying feature function ϕ . In the most extreme case, with no independence assumption at all, evaluating the scoring function would require exhaustive search on the space of all possible assignments of the latent variables. This grows exponentially with the number of latent variables and becomes intractable very quickly.

Tree-structured dependencies are particularly interesting and widely used in practice [1, 4]. Complexity of evaluating the scoring function with tree-structured dependency is quadratic in the cardinality of the domain of a single latent variable.

Dependency structure of the feature function also affects the complexity of finding the fixed latent assignments in inference functions of training, and hence, the construction of the bounds B_t .

Another note is that although we call a_k s positive latent variables and b_k s negative latent variables, this is only for simplicity of notation and explanations. That is, the true meaning of these variables as evidences or counter evidences for the positive class changes by the dependency structure of the scoring function. For example a negative latent variable depending on another negative latent variable of the positive class is not a counter evidence for positive class nor is a evidence. Conversely, positive latent variables dependent to a negative latent variable act as a counter evidence for the positive class.

3 Algorithm

Details of the training framework is given in Section 4 of the main paper. Here is the algorithm corresponding to the training procedure of GLVM. The equation numbers refer to the main paper.

Algorithm 1 Training GLVM classifiers. Note that equation numbers refer to the main paper.

Input: $D = \{(x_i, y_i)\}_{i=1}^n, w_0$

$t \leftarrow 0$

repeat

Construct $\tilde{S}_t(w)$ from Equation (11) and w_t

Construct $\hat{S}_t(w)$ from Equation (12) and w_t

Construct $B_t(w)$ from Equation (8)

$w_{t+1} \leftarrow \arg \min_w B_t(w)$

$t \leftarrow t + 1$

until w_t does not change

Output: w_t

4 Deformable Part Models

Part based models have had a long and successful presence in the history of Computer Vision [2, 3, 4, 5, 9].

Deformable Part Models (DPMs) [4] have been extensively used for object detection over the past decade. Despite their incredible flexibility to object deformation and viewpoint changes they only look for positive evidence in images. In particular, DPMs do not capture

counter evidence. For example, the knowledge that a cow-head detector fires strongly on a detection window should decrease the score of horse-detector on that window. We propose an extension to DPMs by augmenting them with negative parts. We call our model *Generalized-DPM* (or GDPM in short).

Training of the proposed model does not fit in the latent SVM framework [4]. We propose an algorithm to learn DPMs with negative parts. Our experimental results show that negative parts complement DPMs [4]. DPM is a mixture of multi-scale part-based object detectors. Each mixture component captures appearance of the object in a specific viewpoint. Each mixture has a *root* filter for capturing the global structure of the object in a coarse scale. It also has a set of *deformable part* filters for capturing parts of the object individually in a higher resolution. DPM uses sliding window search to detect objects. The root filter is used as the detection window during training. Consider a DPM with n parts. We denote the location (and size) of the root filter and the parts by r and $l = (l_1, \dots, l_n)$ respectively. We also denote the choice of the mixture component by c . The true root, and part locations as well as the choice of the mixture component are unknown even during training and, therefore, are treated as latent variables in a DPM.

We define a *latent configuration* as an assignment of all latent variables in the model and denote it by $z = (c, r, l)$. Score of a DPM with latent configuration z on image x is a linear function of the model parameters, which we denote by β , and is defined as follows:

$$g_\beta(x, z) = \beta \cdot \Phi(x, z) \quad (1)$$

where $\Phi(x, z)$ extracts features from image x according to the latent configuration z .

Let R be a set of candidate root locations (for examples, all detection windows that have high overlap with a given image region). We denote the detection score on R by $f_\beta(x, R)$ and define it as the score of the best latent configuration that places the root filter at a location in R as follows:

$$f_\beta(x, R) = \max_{\substack{z=(r,c,l) \\ s.t. \ r \in R}} g_\beta(x, z) \quad (2)$$

4.1 Training Deformable Part Models

Let $\mathcal{D} = \{(x_i, y_i, b_i)\}_{i=1}^N$ denote a set of N annotated training examples. x_i denotes an image. $y_i \in \{+1, -1\}$ denotes a class label that determines whether the example is foreground ($y_i = +1$) or background ($y_i = -1$). b_i denotes the coordinates of the annotated bounding box that circumscribes the object in a foreground image. b_i is irrelevant if $y_i = -1$. All root filter locations on a background image are considered as independent negative examples.

DPM uses the root filter as a detection window during training. Despite availability of b_i 's for foreground images, DPM treats true detection bounding boxes on the foreground images as latent variables. Any root filter whose overlap with the annotated bounding box b_i exceeds a certain threshold τ is considered as a valid detection (true positive) for b_i . We denote the set of valid detections of b_i by $H(x_i, b_i, \tau)$. In order to train DPMs a separate training set $D = \{(x_i, y_i, R_i)\}_{i=1}^N$ is constructed from the annotated examples in \mathcal{D} . Each background example $(x, y = -1, b_i) \in \mathcal{D}$ adds multiple examples $(x, y, \{r\})$ to D , one for each root placement r . Each foreground example $(x, y = +1, b_i) \in \mathcal{D}$ adds one example (x, y, R) to D where $R = H(x, b, \tau)$. In DPM, the value of τ is set to 0.7.

DPM uses Latent SVM [4] for training. Given the training set D , the model parameters β are trained by minimizing the following objective function:

$$O(\beta) = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N \max\{0, 1 - y_i f_\beta(x_i, R_i)\} \quad (3)$$

The objective function in Equation 3 is non-convex. Particularly, $f_\beta(x_i, R_i)$ as defined in Equation 2 is maximum of a set of linear functions $g_\beta(x, z)$, and therefore is convex. Then for samples with $y > 0$ the hinge loss in Equation 3 becomes concave.

Latent SVM employs an alternating approach similar to CCCP [8] to minimize the training objective. CCCP is an iterative procedure that alternates between 1) constructing a convex upper bound on the original objective function and 2) optimizing the bound. CCCP is guaranteed to converge to a local minimum of the original objective function.

In the case of DPMs, the first step boils down to fixing the value of the latent variables in Equation 3 for *foreground* examples. Let $z_i = (r_i, c_i, l_{i,1}, \dots, l_{i,n})$ denote a particular assignment of the latent variables for the foreground example (x_i, y_i, R_i) . A convex upper bound to $O(\beta)$ is defined as follows:

$$\begin{aligned} B(\beta) = & \frac{1}{2} \|\beta\|^2 + C \sum_{\substack{i=1 \\ y_i=-1}}^N \max\{0, 1 + f_\beta(x_i, R_i)\} \\ & + C \sum_{\substack{i=1 \\ y_i=+1}}^N \max\{0, 1 - g_\beta(x_i, z_i)\} \end{aligned} \quad (4)$$

We call this the *relabeling* step of the training algorithm.

The second step involves solving the following optimization problem:

$$\beta^* = \arg \min_{\beta} B(\beta) \quad (5)$$

Any assignment of z_i 's for foreground examples leads to a convex function (Equation 4) which is an upper bound to O (Equation 3). In order to guarantee progress in each iteration we require that z_i 's are set to the maximum scoring configurations under the previous model β^* as follows:

$$z_i = \arg \max_z g_{\beta^*}(x_i, z), \quad \forall i \text{ s.t. } y_i = +1 \quad (6)$$

4.2 Negative Parts

In a binary classifier, a negative (positive) part is defined as a part whose *maximum* response is *subtracted* from (*added* to) the classification score [6].

While positive parts collect evidence for presence of an object negative parts collect *counter-evidence* for it. For example, consider a detection window ω . Presence of a *saddle* in the detection window ω should decrease the score of a *cow* detector on ω . This is because *cows* tend not to have *saddles* on their backs. Thus, if we find a *saddle* in the detection window of a *cow* detector we, most likely, are confusing a *horse* for a *cow*. Counter evidence is particularly important in discriminating between similar objects; like *cow* from *horse* in

the hypothetical example above. We obtain this behavior by augmenting *cow* detector model with a negative part that fires on *saddle*.

Finally, we note that standard DPMs cannot capture counter-evidence. For example, if F is the appearance model of a positive part, $-F$ does *not* make it a negative part. In this paper we extend DPMs by augmenting them with negative parts.

4.3 Adding Negative Parts to Deformable Part Models

In DPM, responses of all part filters are *added* to the detection score of the classifier. Thus, DPMs cannot model negative parts. We generalize DPMs by augmenting them with negative parts. We call the proposed model *Generalized-DPM* (or GDPM in short). Standard DPM [4] is a GDPM with zero negative parts.

We denote a GDPM with k mixture components by $\gamma = (\gamma_1; \dots; \gamma_k)$. Each mixture component captures the appearance of the object in a viewpoint. A mixture component with n positive parts and m negative parts is parameterized by $\gamma_c = (F_{c,0}; w_{c,1}^+; \dots; w_{c,n}^+; w_{c,1}^-; \dots; w_{c,m}^-)$. $F_{c,0}$ denotes the root filter, w_j^+ 's denote the positive part models, and w_j^- 's denote the negative part models. Each part model $w_{c,j} = (F_{c,j}; d_{c,j})$ has an appearance $F_{c,j}$ and a deformation parameter vector $d_{c,j}$.

Let $l^+ = (l_1^+, \dots, l_n^+)$ denote the positive part placements and $l^- = (l_1^-, \dots, l_m^-)$ denote the negative part placements. We denote a latent configuration in a GDPM by $z = (r, c, l^+, l^-)$. Score of a GDPM γ with latent configuration z on image x is a linear function of the model parameters γ . We denote this by $g_\gamma(x, z)$ and define it as follows:

$$g_\gamma(x, z) = \gamma \cdot \Phi(x, z) \quad (7)$$

where Φ extracts features from image x according to the latent configuration z . This is very similar to the score function of the standard DPM (see Equation 1). For example, both functions are linear in the model parameters. Detection score of placing the root at r , however, is fundamentally different for the two models. For example, unlike Equation 2 we *cannot* compute $f_\gamma(x, R)$ by maximizing over all the latent variables. We will elaborate more on the differences between $f_\beta(x, R)$ and $f_\gamma(x, R)$ in the rest of this section.

Like in DPM, part dependencies in GDPM respect a star configuration with the root filter at the center of the star. Thus, part placements are independent given the location of the root filter. Let $\phi_a(x, l_j)$ denote appearance features extracted from patch l_j of image x (e.g. HOG features). Also let $\phi_d(l_j, r)$ denote the deformation features of placing part j at l_j given that the root filter is placed at r . We define the feature function $\phi(x, l_j, r) = (\phi_a(x, l_j); \phi_d(l_j, r))$. For DPMs, we can rewrite Equation 2 as follows:

$$\begin{aligned} f_\beta(x, R) = & \max_{r \in R} \max_{c=1}^k F_{c,0} \cdot \phi_a(x, r) \\ & + \sum_{j=1}^n \max_{l_j} w_{c,j} \cdot \phi(x, l_j, r) \end{aligned} \quad (8)$$

In contrast, for GDPMs we have:

$$\begin{aligned}
 f_{\gamma}(x, R) = & \max_{r \in R} \max_{c=1}^k F_{c,0} \cdot \phi(x, r) \\
 & + \sum_{j=1}^n \max_{l_j^+} w_{c,j}^+ \cdot \phi(x, l_j^+, r) \\
 & - \sum_{j=1}^m \max_{l_j^-} w_{c,j}^- \cdot \phi(x, l_j^-, r)
 \end{aligned} \tag{9}$$

One way to see that GDPM is more general than DPM is to note that the score function of Equation 8 is convex in the model parameters whereas Equation 9 is not.

4.4 Training with Negative Parts

We define the training objective of a GDPM as follows:

$$O(\gamma) = \frac{1}{2} \|\gamma\|^2 + C \sum_{i=1}^N \max\{0, 1 - y_i f_{\gamma}(x_i, R_i)\} \tag{10}$$

This objective function is non-convex. Moreover, unlike DPMs, fixing the latent variables for the foreground examples does *not* make it convex. Thus, we cannot use Latent SVM framework to train GDPMs.

We deploy CCCP for training the objective function of Equation 10. Recall that CCCP alternates between constructing a convex function that upper bounds the original objective function and optimizing the bound.

We obtain a convex upper bound $B(\gamma)$ to the training objective by replacing $f_{\gamma}(x_i, R_i)$ with a convex upper bound \check{f}_{γ} for background images and a concave lower bound \hat{f}_{γ} for foreground images.

$$\begin{aligned}
 B(\gamma) = & \frac{1}{2} \|\gamma\|^2 + C \sum_{\substack{i=1 \\ y_i=-1}}^N \max\{0, 1 + \check{f}_{\gamma}(x_i, R_i)\} \\
 & + C \sum_{\substack{i=1 \\ y_i=+1}}^N \max\{0, 1 - \hat{f}_{\gamma}(x_i, R_i)\}
 \end{aligned} \tag{11}$$

In the rest of this section we first explain how the bounds are constructed and then elaborate on the optimization of the bound.

4.5 Constructing Convex Bounds

In this section we explain how we obtain \check{f}_{γ} and \hat{f}_{γ} in order to construct the bound $B(\gamma)$ in Equation 11. Let γ^* be the solution to the previous convex bound. We define $l_{i,j}^-(r, c) = \arg \max_l w_{c,j}^{*-} \cdot \phi(x_i, l, r)$ to be the optimal placement of the j -th negative part using the previous model conditioned on the choice of the root location r and the mixture component c .

We obtain \check{f}_γ by fixing the placement of the negative parts according to $l_{i,j}^-(r, c)$ as follows:

$$\begin{aligned} \check{f}_\gamma(x_i, R_i) = & \max_{r \in R} \max_{c=1}^k F_{c,0} \cdot \phi(x_i, r) \\ & + \sum_{j=1}^n \max_{l_j^+} w_{c,j}^+ \cdot \phi(x_i, l_j^+, r) \\ & - \sum_{j=1}^m w_{c,j}^- \cdot \phi(x_i, l_{i,j}^-(r, c), r) \end{aligned} \quad (12)$$

Let (r_i, c_i, l_i^+) denote the best placement of the root filter, choice of the mixture component, and placement of the positive parts using the model γ^* given that the negative parts have been placed optimally according to $l_{i,j}^-(r, c)$. We can formally define this as follows:

$$\begin{aligned} (r_i, c_i, l_i^+) = & \arg \max_{(r, c, l)} F_{c,0}^* \cdot \phi(x_i, r) \\ & + \sum_{j=1}^n w_{c,j}^{*+} \cdot \phi(x_i, l_j, r) \\ & - \sum_{j=1}^m w_{c,j}^{*-} \cdot \phi(x_i, l_{i,j}^-(r, c), r) \end{aligned} \quad (13)$$

We obtain \hat{f}_γ as follows:

$$\begin{aligned} \hat{f}_\gamma(x_i, R_i) = & F_{c_i,0} \cdot \phi(x_i, r_i) \\ & + \sum_{j=1}^n w_{c_i,j}^+ \cdot \phi(x_i, l_{i,j}^+, r_i) \\ & - \sum_{j=1}^m \max_{l_j^-} w_{c_i,j}^- \cdot \phi(x_i, l_j^-, r_i) \end{aligned} \quad (14)$$

4.6 Initializing Negative Parts

The positive parts in original DPM [4] is initialized by first learning a root filter (whole object detector without parts). Then in order to initialize the location and weights of a part filter, a submatrix of root filter is sought which has the maximal positive energy. Positive energy is measured by sum of squared *positive* weights in the submatrix of the root filter. The first part is initialized with the best submatrix location and weights. Then all the weights are zeroed at that location and the procedure is repeated to initialize a second part. This continues until the desired number of parts are added. We initialize positive parts in the same manner. Also, we initialize the negative part similarly, except that we try to find the submatrix with highest negative energy being sum of squared *negative* weights of the submatrix. This involves the main results of the paper.

We further initialize the negative parts with the positive parts of similar classes. Particularly, we initialize the negative parts for *cow* GDPM model by taking positive parts from *horse* DPM model.

4.7 Inference for Data Mining

When doing inference for GDPM data mining, we need to construct a concave lower bound for positive samples and convex upper bound for negative samples. That is, we need to fix the position of negative parts for negative samples at the location of relabeling. We further need to fix the location of positive parts, mixture component and root filter location for positive samples to the assignments at the relabeling. On the other hand, the positive parts and mixture component needs to be maximized for negative samples and negative parts needs to be updated for positive samples. Storing the fixed values of latent variables for all negative samples (hundreds of thousands negative bounding box for each image) is prohibitively memory-consuming. Thus, in our implementation, we do the following. At each relabeling step we keep a copy of the model. We also store the inferred component and location of the root filter only for positive samples (requires low amount of memory since number of positive annotations are usually low). Then at each iteration of data mining: For positive samples we take the current model and roll back the positive parts to the copied model, and do the inference at the saved location and with the saved component. As for negative examples, we take the current model and roll back the negative parts to the copied model and do the inference. This way we simply take care of convex upper bound and concave lower bounds with low memory footprint. However, this comes at the cost of two times more computational complexity. Since the main complexity of the inferences goes back to convolution of part/root filters, the remedy for this would be to save the output of convolution for positive parts on positive examples and negative parts on negative examples (proportional to the number of images and independent of number of negative bounding boxes per image).

5 Dataset

Random images from Cat Head dataset [7]. Note that, many of the cat heads are not frontal.

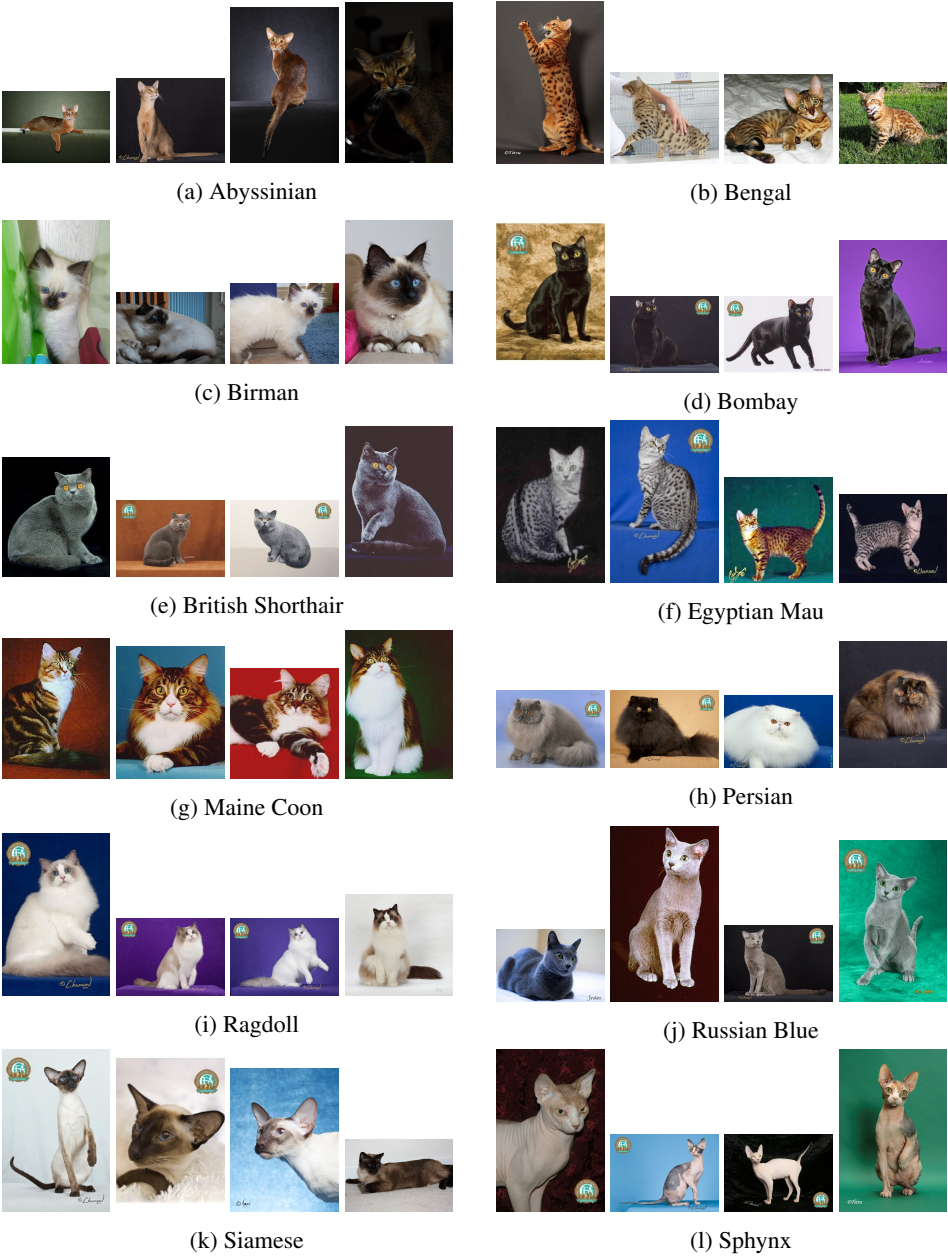
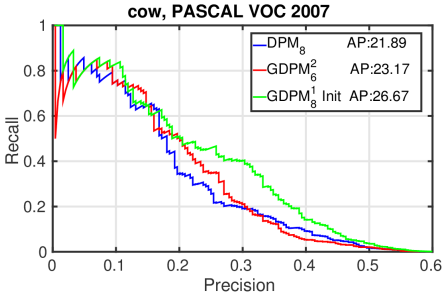


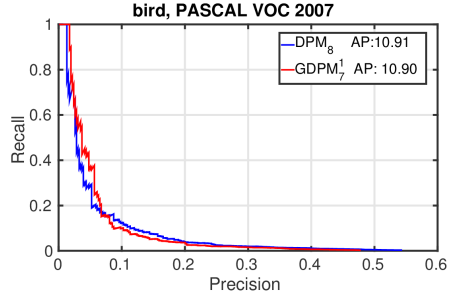
Figure 2: Samples of Pet [7] dataset used for cat head detection

6 Precision Recall Curves

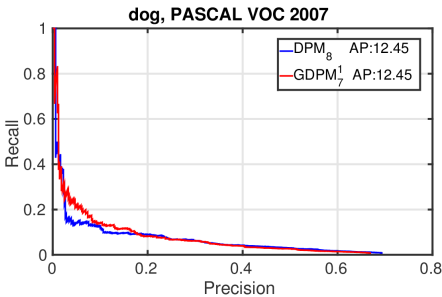
In this section, the precision recall curves for DPM comparing with its best performing GDPM variant is plotted for animals subset of PASCAL VOC 2007. For the class of *cow* an extra curve is plotted which belongs to a different initialization of negative parts using *horse* DPM model. For the class of *bird* and *dog* while the AP (calculated by average of 11 sampled points) is equal for GDPM and DPM, the GDPM curve shows slightly better precision at lower recalls.



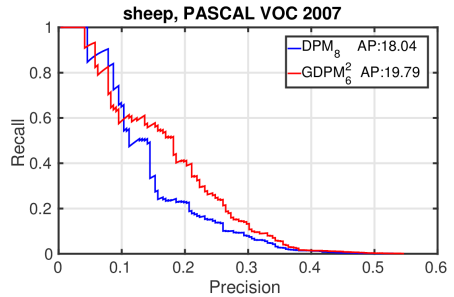
(a) PR curve for *cow* in PASCAL VOC 2007 using original DPM, GDPM with 2 negative parts, and GDPM with one negative part initialized by horse DPM model.



(b) PR curve for *bird* in PASCAL VOC 2007 using original DPM, GDPM with 1 negative part. Although the AP (computed by taking average of 10 sampled points from the curve) is equal, the GDPM curve has slightly higher precision at lower recall.



(c) PR curve for *dog* in PASCAL VOC 2007 using original DPM, GDPM with 1 negative part. Although the AP (computed by taking average of 10 sampled points from the curve) is equal, the GDPM curve has slightly higher precision at lower recall.



(d) PR curve for *sheep* in PASCAL VOC 2007 using original DPM, GDPM with 2 negative part. The GDPM curve exhibits higher precision at different recalls.

Figure 3: PR curves for animals subset of PASCAL VOC 2007

References

- [1] Hossein Azizpour and Ivan Laptev. Object detection using strongly-supervised deformable part models. In *ECCV*, 2012.
- [2] Lubomir D. Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, pages 1365–1372, 2009.
- [3] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Comparing active shape models with active appearance models. In *BMVC*, pages 1–10, 1999.
- [4] Pedro F. Felzenszwalb, David A. McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.
- [5] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77(1-3):259–289, 2008.
- [6] Sobhan Naderi Parizi, Andrea Vedaldi, Andrew Zisserman, and Pedro F. Felzenszwalb. Automatic discovery and optimization of parts for image classification. In *ICLR*, 2015.
- [7] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [8] Alan Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.
- [9] Long Zhu, Yuanhao Chen, and Alan L. Yuille. Learning a hierarchical deformable template for rapid deformable object parsing. *PAMI*, 32(6):1029–1043, 2010.